Algorithms and Data Structures 2
CS 1501

Fall 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

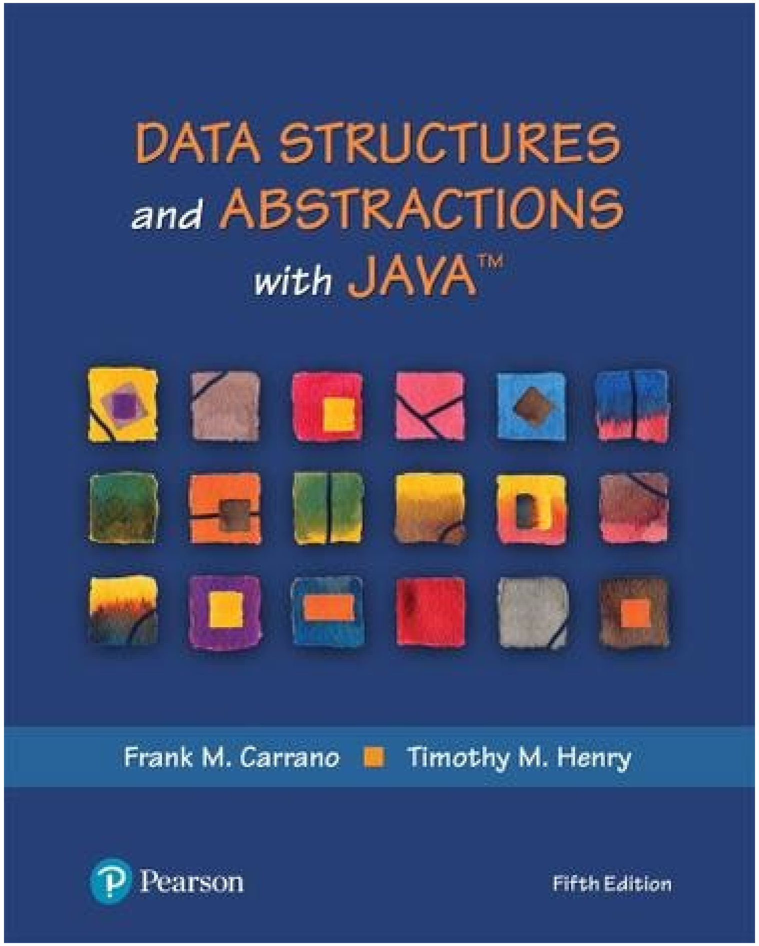
Contact Info

- **Course website:** http://www.cs.pitt.edu/~skhattab/cs1501/
- **Instructor:** Sherif Khattab ksm73@pitt.edu
- **My Student Support Hours:** https://khattab.youcanbook.me
  - MW: 10:00-12:00; TuTh: 13:00-15:00; F by appointment
  - 6307 Sennott Square, Virtual Office: https://pitt.zoom.us/my/khattab
  - Please schedule at: https://khattab.youcanbook.me/
- **Teaching Team:**
  - Junshang Jia, juj22@pitt.edu
  - Christofer Hinson, chh183@pitt.edu
  - Connor Sweeney, cps43@pitt.edu
  - More TAs to come
- No recitations this week, but you got some work to do!
- **Communication**

  Piazza (**Please expect a response within 72 hours)**

  Email not recommended!

Textbooks



Algorithms (4th Edition)

Robert Sedgewick and Kevin Wayne

Online Resources: https://algs4.cs.princeton.edu/

Data Structures and Abstractions with Java (5th Edition)

Frank M. Carrano and Timothy M. Henry

Grades

- 40% on best four out of five programming assignments; mostly autograded
  - posted on Canvas, distributed using Githib, and submitted on **Gradescope** from Github
- 20% on homework assignments on Gradescope
- 20% on exams: 12% on higher grade and 8% on lower
- 10% on lab exercises; mostly autograded
- 10% on in-class Top Hat questions

Canvas Walkthrough

- Lectures posted on Tophat
  - Draft slides available on Github
- Lecture and recitation recordings
  - under Panopto Video
- RedShelf Inclusive Access for the Sedgewick Textbook
  - You can cancel before Add/Drop
- Piazza for discussion and communication
- Gradescope and autograding policies
- Academic Integrity
- NameCoach

Expectations

- Your continuous feedback is important!
  - Anonymous Qualtrics survey
  - Midterm and Final OMET
- Your engagement is valued and expected with
  - classmates
  - teaching team
  - material

Lecture structure (mostly)

Time	Description
~5 min before and after class	Informal chat
~25 min	Announcements, review of muddiest points on previous lecture, and QA on assignments/labs/homework problems
~45 min	Lecturing with Tophat questions and/or activities
~5 minutes	QA and muddiest points/reflections

Why is this class (notoriously) hard?

- Lots of concepts
  - Attend lectures and recitations (if you absolutely cannot attend, watch the video recordings)
  - Study often!
  - Put effort into the weekly homework assignments
- Programming Assignments are relatively hard
  - Refresh your Java programming (CS 0445) and **<u>debugging skills</u>**
  - Start early and show up to student support hours!

Goals of the course

- To convert non-trivial *algorithms* and data structures into *programs*
  - Various issues will always pop-up during *implementation*
    - Such as?...
- To analyze algorithms and how they affect the run-times of the associated programs
  - Different solutions can be compared using many metrics

Announcements

- Lab 0 is due this Friday (not graded)
- Recitations start next week
- Homework 1 will be assigned this Friday
- JDB Example will be available on Canvas
- Draft slides and handouts available on Canvas

Today's Agenda

- A technique for modeling runtime of algorithms
  - $\sum_{all\ statements} Cost * frequency$
- Determining the order of growth of a function
  - Ignoring lower-order terms and multiplicative constants
  - The Big O family

Let’s consider the ThreeSum problem

- 3-sum Problem
  - Given a set of arbitrary integers find out how many **distinct** triples sum to exactly zero
  - do you have questions on the problem specification?
- Example input:
  - 5, -1, 2, -3, -2, 1, 0
  - what should the output be?

Brute-force solution

Enumerate all possible distinct triples and check their sums

```
cnt = 0
for each distinct triple
    if sum of triple equals zero
        increment cnt
```

- How would you enumerate all distinct triples?
- What if all the input integers are unique?

Brute-force solution

```
public static int count(int[] a) {
    int n = a.length;
    int cnt = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            for (int k = j+1; k < n; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    cnt++;
                }
            }
        }
    }
    return cnt;
}
```

- Why is it correct to start the j loop from i+1?
  - Would we miss a triple if we do so?
  - Is it correct to start the j loop from 0?

ThreeSum: brute-force, 3-loop solution

```
public static int count(int[] a) {
    int n = a.length;
    int cnt = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            for (int k = j+1; k < n; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    cnt++;
                }
            }
        }
    }
    return cnt;
}
```

Would that solution be correct if the input integers are not distinct?

Mathematically modelling runtime

- Ru

What is the runtime?

A technique for modeling runtime of algorithms

- $\sum_{all\ statements} Cost * frequency$

- Split the algorithm into blocks such that
  - the code statements in each block have the same frequency
- $\sum_{all\ blocks} Cost * frequency$

Algorithm Analysis Example 1

$$\text{for}\left(\overset{1}{\boxed{i=0}};\ \overset{n+1}{\boxed{i<n}};\ \overset{n}{i{+}{+}}\right)$$

$$a[i] = i;$$

Algorithm Analysis Example 2

1

0 or 1

0 or n

```
if( x >0 ){
  for( i=0; i<n; i++)
    a[i] = i;
```

Algorithm Analysis Example 3

1

```
for ( i = n ; i >= 1 ; i = i/2 )
    a[i] = i;
```

log n

What is the runtime?

```
public static int count(int[] a) {
    int n = a.length;
    int cnt = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i+1; j < n; j++) {
            for (int k = j+1; k < n; k++) {
                if (a[i] + a[j] + a[k] == 0) {
                    cnt++;
                }
            }
        }
    }
    return cnt;
}
```

A couple useful Math formulae

$\sum$ arithmetic Series:

$1 + 2 + 3 + 4 + \cdots + n$

$= \#\text{terms}\left(\frac{\text{first term} + \text{last term}}{2}\right)$

$= \Theta(\#\text{terms} * \text{largest term})$

$\sum$ geometric Series:

$1, 2, 4, 8, 16, \ldots$

$1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$

$= \Theta(\text{largest term})$

Runtime Analysis of 3-loop Algorithm for ThreeSum

1

```
for(i=0; i<n; i++)
    for(j=i+1; j<n; j++)
        for(k=j+1; k<n; k++)
            if(a[i]+a[j]+a[k]==0)
                cnt++;
```

n

n^2 $\quad$ n^3 $\quad$ $0 \ldots n^3$

Best Case: $1+n+n^2+n^3+0=\Theta(n^3)$

Worst Case: $1+n+n^2+n^3+n^3=\Theta(n^3)$

Enter Asymptotic Analysis

Algorithm Analysis

- Determine *resource usage* as a function of *input size*
- Measure ***asymptotic*** performance
  - Performance as input size increases to infinity

Asymptotic performance

Focus on the **order of growth** not on exact values

- How fast the function value increases when the input increases

Common orders of growth

- Constant - 1
- Logarithmic - log n
- Linear - n
- Linearithmic - n log n
- Quadratic - n^2
- Cubic - n^3
- Exponential - 2^n
- Factorial - n!

Side note

What does $log_2 n$ really mean?

Quick algorithm analysis

- How can we determine the order of growth of a function?
  - Ignore lower-order terms
  - Ignore multiplicative constants
- Warning: this is a simplification. It works for most of the algorithms in this course.
- In some cases, it is difficult to determine the highest-order term
- In some cases, the constant factors play a significant role
  - e.g., small or medum-size input and large constant factors

Example

$$5n^3 + 53n + 7 \ \rightarrow n^3$$

- Can we say $5n^3 + 53n + 7 = n^3$?
- No! We need a mathematical notation
- $5n^3 + 53n + 7 = O(n^3)$
- It means the order of growth of $5n^3 + 53n + 7$ is no more than ($\leq$) the order of growth of n^3

The Big O Family

- O roughly means $\leq$
  - Big O
- o roughly means $<$
  - Little O or O-micron
- Ω roughly means $\geq$
  - Big Omega
- ω roughly means $>$
  - Little Omega
- Θ roughly means $=$
  - Theta
- Relationships are between orders of growth, not between exact values!

Notations

- May also see:
  - $f(x) \in O(g(x))$ or
  - $f(x) = O(g(x))$
- used to mean that $f(x)$ is $O(g(x))$
- Same for the other functions

Notations

$O(n)$

$10n$

$\log n$

$50n+1$

7

$3\sqrt{n}+10$

n^2

$10n \in O(n)$

$\log n \in O(n)$

$n^2 \notin O(n)$

Theta vs. Tilde

Tilde approximation (~)

- Same as Theta but keeps constant factors
- Two functions are Tilde of each other is they have the same order of growth and the same constant of the largest term

$$5n = \Theta(5{,}000{,}000{,}000\ n)$$

$$\not\sim 5{,}000{,}000{,}000\ n$$

Wait…

- Assuming that definition…
  - Is ThreeSum $O(n^4)$?
  - What about $O(n^5)$?
  - What about $O(3^n)$??
- If all of these are true, why was $O(n^3)$ what we jumped to to start?

A faster algorithm for 3-sum

- What if we sorted the array first?
  - Pick two numbers, then binary search for the third one that will make a sum of zero
    - e.g., a[i] = 10, a[j] = -7, binary search for -3
    - Still have two for-loops, but we replace the third with a binary search
      - Runtime now?

$$\underbrace{n^2}_{\text{all pairs}} \underbrace{\log n}_{\text{Binary Search}} + \cancel{\underbrace{n \log n}_{\text{Sorting}}}$$

    - What if the input data isn't sorted?
- What about the sorting time?

The 3-sum problem: can we do better?

- There is an O(n^2) algorithm
- There is also an O(n log n) algorithm under special cases
- **Unsolved problem**: Is there an O($n^{2-\varepsilon}$) algorithm for some $\varepsilon > 0$?